CS 189: Introduction to Machine Learning Spring 2014

Homework 6

He Ma SID: 22348372
Yina Jin SID: 21921178

Data Pre-processing

1. Images
   The images are parsed to a $n \times (d+1)$ matrix with n as the number of images and d as the 28×28 features. Each image is normalized by l_2 norm as described in HW3. To ease the calculation, a column of all 1s is added to the front to for the bias.

2. Labels
   The labels are parsed to a $n \times 10$ matrix. Each row contains a 1 at index $(label + 1)$, and 0s for the rest.

Single Layer Neural Network: Mean Squared Error

1. Stochastic gradient update
   For bias terms, we plan to have a bias term on each layer of the neural network. In this case we have only 1 layer, which means we have a 1 as the 1st neuron in the base layer. Thus the bias will be included in the weight updates. Also no hidden layers means there's only base step in this case:

$$\delta_i^1 = \frac{\partial e(w)}{\partial S_i^1} = \frac{\partial}{\partial S_i^1}(\frac{1}{2}(\sigma(S_i^1) - y_i)^2) = (\sigma(S_i^1) - y_i) \times \sigma\prime(S_i^1)$$

   In Matlab, we can write it in vector form:

$$W = W - \eta X^0 {\delta^1}^T = W - \eta X^0((\sigma(S^1) - y). * \sigma\prime(S^1))^T$$

2. Parameters
   Mini-batch size: 200
   Number of epochs: 400
   Initial weights: all weights are uniformly distributed between -1 and 1
   Learning rate: 10^{-3} for all epochs

3. Result
   Running time: 4841.986 seconds (40 minutes)

   Best training error: 0.0860
   Best test error: 0.0835

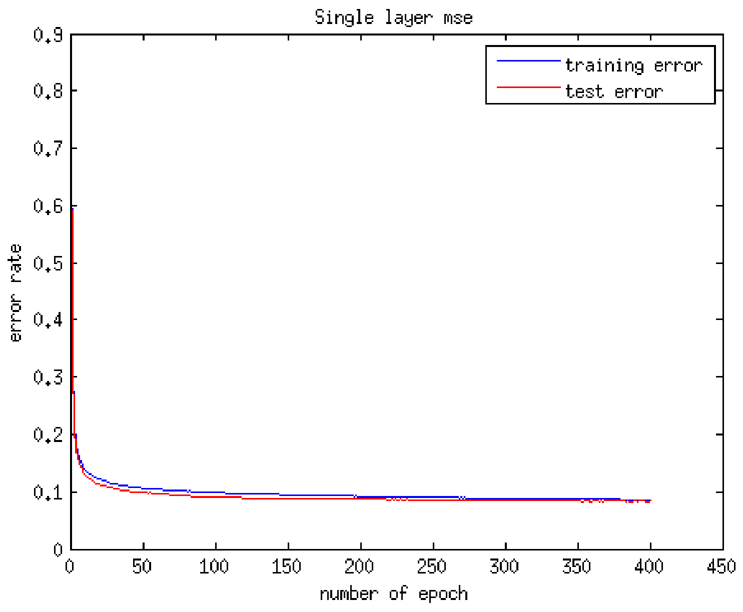

Figure 1: Single Layer Neural Network with loss function MSE)

As we can see from the plot, the error rate for both training and test dataset are decreasing. If more epochs are used, we could achieve a slightly better result.

Single Layer Neural Network: Cross Entropy Error

1. Stochastic gradient update For bias terms, we plan to add a bias term to each layer. In this case we only have 1 layer, which means we have a 1 as the 1st neuron in the base layer.

   $$\delta_i^1 = \frac{\partial e(w)}{\partial S_i^1} = \frac{\partial}{\partial S_i^1} - [y_i \ln(\sigma(S_i^1)) + (1-y_i)\ln(1-\sigma(S_i^1))] = -[\frac{y_i}{\sigma(S_i^1)} - \frac{1-y_i}{1-\sigma(S_i^1)}] \times \sigma\prime(S_i^1)$$

   In Matlab, we could write it in vector form:

   $$\delta^1 = -[y./\sigma(S^1) - (1-y)./(1-\sigma(S^1))] \times \sigma\prime(S^1)$$

   So

   $$W = W - \eta X^0 \delta^T = W - \eta X^0 {\delta^1}^T$$

2. Parameters
   Mini-batch size: 200
   Number of epochs: 400
   Initial weights: all weights are uniformly distributed between -1 and 1
   Learning rate: 10^{-3} for all epochs

3. Result
   Running time: 5008.216 seconds (43 minutes)
   Best training error: 0.0764
   Best test error: 0.0784

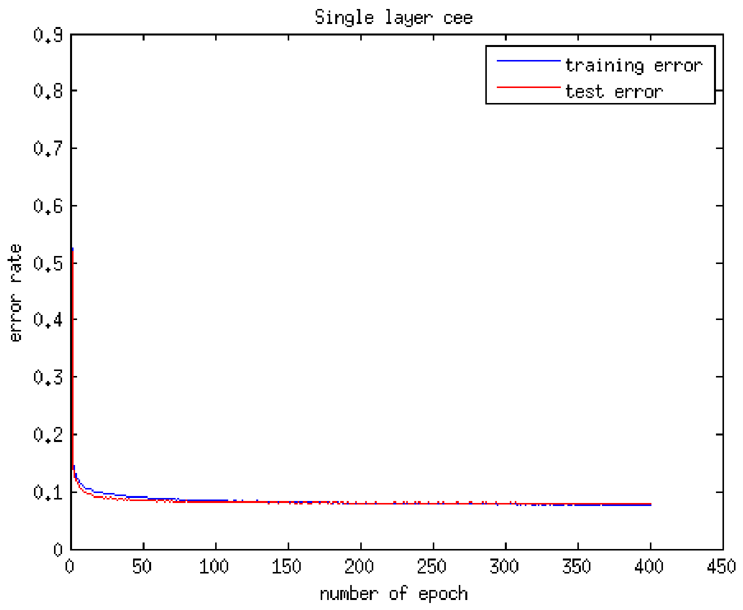

Figure 2: Single Layer Neural Network with loss function CEE)

As we can see from the plot, the error rate for both training and test dataset are decreasing. If more epochs are used, we could achieve a slightly better result.

Multilayer Feed Forward Neural Network: Mean Squared Error

1. Stochastic gradient update
   Remember we decide to include the bias term within the neuron networks structure. So the bias updates is included in the weight updates. The base case should be the same as the single layer. For lower layers (inductive step),

$$\delta_i^{l-1} = \frac{\partial e(w)}{\partial S_i^{l-1}} = \sum_j \frac{\partial e(w)}{\partial S_j^l} \frac{\partial S_j^l}{\partial x_i^{l-1}} \frac{\partial x_i^{l-1}}{\partial s_i^{l-1}} = \sum_j \delta_j^l w_{ij}^l \tanh'(s_i^{l-1}) = (1-(\tanh(s_i^{l-1}))^2) \sum_j w_{ij}^l \delta_j^l$$

   In matlab the vectorized form of data can be written as

$$\delta^{l-1} = (1 - (\tanh(s^{l-1})). * (\tanh(s^{l-1}))). * (W^l * \delta^l)$$

   Then the weight updates would be

$$W = W - \eta X \delta^T$$

2. Parameters
   Mini-batch size: 200
   Number of epochs: 11
   Initial weights: all weights are uniformly distributed between -1 and 1
   Learning rate: 10^{-4} for all epochs

3. Result Running time: 1hr
   Best training error: 0.1218
   Best test error: 0.1157

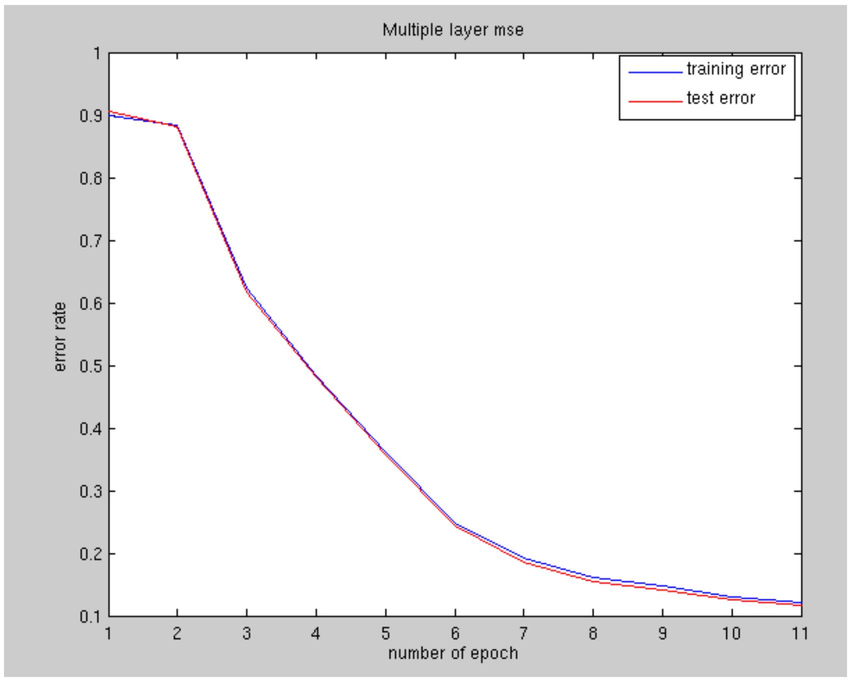

Figure 3: Multiple Layer Neural Network with loss function MSE)

We should be able to get a better result by adding more epochs.

2 Multilayer Feed Forward Neural Network: Cross Entropy Error

1. Stochastic gradient update Again the bias updates is included in the weight updates, and the base case should be the same as the single layer. For lower layers (inductive step), it's exactly the same with Mean Squared Error because the difference lies only on the base case.

2. Parameters
   Mini-batch size: 200
   Number of epochs: 11
   Initial weights: all weights are uniformly distributed between -1 and 1
   Learning rate: 10^{-4} for all epochs

3. Result Running time: 1hr
   Best training error: 0.1068
   Best test error: 0.1034

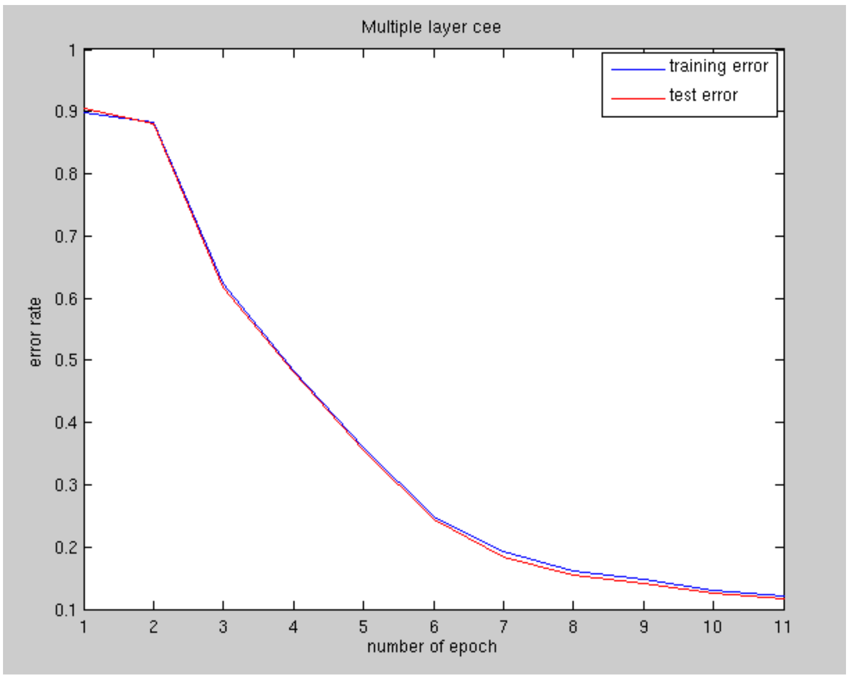

Figure 4: Multiple Layer Neural Network with loss function CEE)

We should be able to get a better result by adding more epochs.

Reference

1. Abu Mostafa's lecture video and slides
2. Malik's lecture notes
3. For our sigmoid function: http://cs229.stanford.edu/section/matlab/sigmoid.m